AERONAUTICAL SYSTEMS CENTER MAJOR SHARED RESOURCE CENTER

SGI ALTIX 3700 USER'S GUIDE

November 2, 2005

Release Notes

Last reviewed on November 2, 2005

Issued November 2, 2005

Corrected an misspelling in the pam -auto_place option.
Added information acout current memory limits.

Issued September 28, 2005

Initial release.

Issued Spetember 21, 2005

Draft release. Updated information related to running MPI, SHMEM and hybrid codes.

Issued July 11, 2005

Added #BSUB -a [SMP|MPI|MIX] information.

Table of Contents

1. Introduction

This document provides an overview and introduction to using the SGI Altix 3700 pioneer system, located at the Aeronautical Systems Center (ASC) Major Shared Resource Center (MSRC). The ASC MSRC is located at Wright-Patterson Air Force Base, near Dayton, Ohio. This guide is intended to provide information so that customers who are familiar with the UNIX operating system can create and run their own programs, as well as use existing application software on the SGI Altix 3700 system.

1.1 Assumed Background of the Reader

It is assumed that the reader of this guide has a firm grasp of the concepts required to use the UNIX operating system and to program in either the C, C++, FORTRAN 77, FORTRAN 90, or FORTRAN 95 languages. It is also assumed that the reader has read the *ASC MSRC User's Guide*, which contains site specific information about the ASC MSRC. The *ASC MSRC User's Guide* is available from the ASC MSRC Service Center at 1-888-MSRC-ASC (1-888-677-2272), (937)255-0194, or DSN 785-0194. It is also available in PostScript and PDF formats as described below in Section 1.7.

1.2 Hardware Overview

The SGI Altix 3700 system is a Shared Memory system with 512 Central Processing Units (CPUs) per node. Each CPU is a 1.6 GigaHertz (GHz) Itanium2 processor and has a 32 kilobyte (KB) of Level 1 cache, 256 KB of Level 2 cache, 8 megabytes (MB) of Level 3 cache and access to 1GB of system memory. However, due to Linux memory management, only 900MB of each 1GB is user-accessible. We are working with SGI to improve this.

1.3 Accessing the System

While the SGI Altix 3700 has a total of 2048 CPUs, divided into 4 nodes with 512 CPUs per node, only 500 CPUs per node are available for user jobs. 12 CPUs per node are reserved for OS-use and are not accessible by the user. The fully qualified hostname of the interactive system is *eagle.asc.hpc.mil*.

Users are only permitted to login onto the interactive node of the system. The other nodes are for batch jobs only. Users submit their jobs on the front-end node and the batch system will automatically start their jobs on the other systems based on the load of the system. See the *ASC MSRC User's Guide* for instructions on accessing the machines.

1.4 ASC MSRC Connectivity

Since the SGI Altix 3700 is an integrated component of the ASC MSRC, user files are Network File System (NFS) mounted from the ASC MSRC High Availability File Server (HAFS) system to the system. When users log into a system, their home ($HOME) directory (which will be the current directory immediately after logging in) physically resides on the file server, but appears to be local to the system. The ASC MSRC also supplies archival storage and visualization capabilities.

1.5 ASC MSRC Startup Files

All users are provided a `.cshrc` and `.login` file in their home directory. These

files reference standard setup files, maintained by the site administrators in a central location, which set up a standard environment for all MSRC users. These files **should not** be modified.

To set up specific information for your SGI Altix 3700 session, such as environment variables, path information, terminal information, or command aliases, place the appropriate commands and information into files called `.personal.cshrc` and `.personal.login`. The standard startup files check your home directory for the existence of these files and executes them if found. Commands related to aliases, prompts, and some environment variables should go into `.personal.cshrc`, while commands related to the type of terminal you are using should go into `.personal.login`. See Section 3 of this guide for more details on the computing environment and the *ASC MSRC User's Guide* for more details on startup files.

1.6 The Archive Command

The archive system ($ARC) is mounted on the interactive node of the system, however, to improve transfer speeds for batch jobs, $ARC **is not** mounted to the batch nodes. To transfer files from $ARC, users will have to use un-kerberized rcp or the archive command.

The **archive** command is a recently added tool to the ASC MSRC to help users with transferring files to and from $ARC. The basic syntax for the archive command is:

archive get [getopts] file1 [file2 ...]
archive put [putopts] file1 [file2 ...]

More information on the **archive** command can be found online via the ***archive*** man page (***man archive***).

1.7 Additional Information

Much of the information presented in this document is available online through the man pages and is accessible by typing:

man {command name}

when logged into *eagle-0*.

The *ASC MSRC User's Guide* and this document are all available in PostScript and PDF format. They may be downloaded via the ASC MSRC website at

```
http://www.asc.hpc.mil/customer/userdocs
```

2. ASC MSRC SGI Altix 3700

This section details the hardware and software available on the SGI Altix 3700 and how they are currently configured.

2.1 Hardware

The SGI Altix 3700 has a total of 2048 CPUs. Each CPU is a 1.6GHz Itanium2 processor with a peak speed of 5.7 GFLOPS, providing a total capacity of approximately 11.64 TFLOPs. Each CPU has a primary data cache of 32 KB, a primary instruction cache of 32 KB, an on-board cache of 8 MB and access to 1GB of system memory. At this time, the system is divided into 4 separate nodes. The system has a Symmetric MultiProcessing (SMP) architecture with 512 CPUs , 512 GB memory per node and a total of 40TB of disk space. 500 CPUs per node are available for user jobs, 12 CPUs per node are reserved for operating system functions.

The interactive node of eagle (eagle-0), has the same specifications as above, but is limited to 16 CPUs.

2.2 File System Overview

Diskspace is subdivided into several areas:

- System space (i.e., `/usr`, `/opt`.)
- `/work1`
- `/work2`

2.2.1 Workspace

Workspace is a filesystem local to each machine that batch jobs are required to run their jobs in. Workspace on the SGI Altix totals 100 TB, however, is divided into two 43 TB cabinets for administrative and performance reasons. Upon your first login, you will be assigned a workspace directory on either `/work1` or `/work2` depending on whether your UID (you can get your UID from the unix ***id*** command) is even or odd. Once assigned, your $WRK variable will always point to either `/work1/<username>` or `/work2/<username>`.

I/O on the `/work1` and `/work2` filesystems are quicker than I/O on a file system mounted from the network (such as $HOME or $ARC). `/workspace` is intended for the **temporary** storage of data files needed for your application. This includes (but is not limited to) grid files, restart files, input files, and output files. $WRK is to be used rather than the `/tmp` and `/usr/tmp` directory areas to prevent possible system crashes.

There is no quota on the amount of disk space you may use in workspace, but a file scrubber is used to automatically remove old files to prevent it from becoming filled. The current policy for removing files is on the ASC MSRC web page and is subject to change based on periodic reviews.

The $WRK file system is NOT backed up. In the event of deletion or catastrophic media failure, files and data structures are lost. It is your

responsibility to transfer files that need to be saved to a location that allows permanent storage such as $HOME or, preferably, $ARC.

2.3 Operating System

The SGI Altix 3700 runs a version of Red Hat Enterprise Linux AS release 3 modified by SGI to include SGI extensions.

2.4 Available Software

Software currently available on the SGI Altix 3700 includes: the Intel and GNU FORTRAN, C, and C++ compilers; MPI and many third party software packages.

3. Program Development

Program development in the SGI Altix 3700 computing environment is similar to that used in a typical UNIX environment. However, the user must take additional steps to utilize the multiple processors available.

3.1 Development Tools

The ASC MSRC offers many tools to help users who write their own code to develop, compile and debug their software.

The SGI Altix 3700 implements the Intel Fortran, C and C++ compilers, along with the GNU Fortran, C and C++ compilers. The Intel FORTRAN compiler is a FORTRAN 90 compiler, whereas the GNU g77 compiler is a FORTRAN 77 compiler.

The SGI Scientific Computing Software Library (SCSL) is installed, which provides a collection of mathematical and scientific libraries including Basic Linear Algebra Subprograms (BLAS) levels 1, 2, and 3; LAPACK; Fast Fourier Transforms (FFTs); and convolutions.

To aid in debugging your software applications, the ASC MSRC provides the Intel Debugger and the GNU Debugger.

The Intel Debugger and GNU Debugger are symbolic source code debuggers that debug programs compiled by the Intel(R) C/C++ Compiler, the Intel(R) Fortran Compiler, and the GNU compilers (gcc, g++). For full source-level debugging, compile the source code with the compiler option that includes the symbol table information in the compiled executable file. Intel IDB supports DBX and GDB modes. In the GDB mode, Intel Debugger operates like the GNU Debugger, GDB.

Documentation for these compilers, libraries, and tools is available online in the man page by executing a man on ***ifort***, ***icc***, ***icpc***, ***gcc***, ***g++***, ***g77***, ***scsl***, ***gdb*** or ***idb***

3.2 Parallel Processing

Users may utilize multiple CPUs to execute their programs. The compilers are capable of creating parallel programs through the use of compiler directives and parallel standards such as Message Passing Interface (MPI) and OpenMP.

3.2.1 MPI

The goal of MPI is to develop a widely used standard for writing message-passing programs. As such, the interface attempts to establish a practical, portable, efficient, and flexible standard for message passing.

You can compile MPI programs on the SGI Altix 3700 using the following command line options.

For MPI FORTRAN codes:

ifort -o prog prog.f -lmpi

For MPI C/C++ codes:

icc -o prog prog.c -lmpi

icpc -o prog prog.c -lmpi -lmpi++

Other compiler options are available. Please consult the man pages for the compiler you are using for more information.

To execute MPI code on the SGI Altix 3700 you must use the ***pam*** or ***mpirun*** command.

pam -mpi -a eagle -auto_place {My Program}

As you've probably noticed, there is no need to give pam the number of cpus you need to run (the -np option in mpirun). This is because pam is designed to work closely with LSF and is given the number of cpus to use from the #BSUB -n option in your LSF script.

More information on MPI can be obtained from:

`http://www.mpi-forum.org`

3.2.2 OpenMP

OpenMP is a specification for a set of compiler directives, library routines, and environment variables that can be used to specify shared memory parallelism in FORTRAN and C/C++ programs.

Creating an OpenMP program is done through OpenMP directives in the source code and by adding the -omp flag to your compile string.

To run an OpenMP program, you must first tell the program how many threads (processors) to use. This is achieved through the OMP_NUM_THREADS environment variable. To set this variable, use the following command in *csh*:

setenv OMP_NUM_THREADS x

where *x* is the number of CPUs you wish to run on.

To execute OpenMP code on the SGI Altix 3700 you must use the dplace command.

dplace -x2 -c{cpu range} {My Program}

The cpu range is the range of cpus you need to run, starting with 0. For example, if you needed 16 cpus to run your job, your cpu range would be 0-15, 0-63 for 64 cpus, etc.

For more information on OpenMP and its directives, please see the following page:

`http://www.openmp.org`

3.2.3 Shared Memory or SHMEM

Shared Memory, or SHMEM, is a set of libraries that achieve parallelization in user's code using data passing or one-sided communication techniques. This method of parallelism requires that all processes and threads have access to the same pool of memory.

To execute SHMEM code on the SGI Altix 3700 you must use the dplace command.

dplace -x2 -c{cpu range} {My Program}

The cpu range is the range of cpus you need to run, starting with 0. For example, if you needed 16 cpus to run your job, your cpu range would be 0-15, 0-63 for 64 cpus, etc.

For more information on SGI's Altix SHMEM implementation, please see the following page:

http://www.sgi.com/developers/technology/gsm.html

3.2.4 MPI/OpenMP Hybrid Code

Some users are experimenting with codes implementing a hybrid of MPI and OpenMP calls. While the SGI Altix 3700 can run such code, there are some issues and limitations with doing so.

The OpenMP threads of a hybrid code all need access to the same memory pool. Because of this, hybrid applications are unable to run across more than one node. Unfortunately, hybrid codes will not be able to run on more than 500 cpus on the Altix.

Since hybrid codes cannot cross nodes, users must use the mpirun command to execute their code since pam may try to execute their code across nodes. The syntax for the SGI Altix 3700's mpirun command is similar to other platforms users are used to running on. Users must also set the OMP_NUM_THREADS variable, much like in the OpenMP section above.

setenv OMP_NUM_THREADS {Number of threads}
mpirun -np {number of cpus} {My program}

3.3 FORTRAN Programming

The default FORTRAN compiler on the SGI Altix 3700 is the Intel FORTRAN compiler. The FORTRAN compiler commands are f77 and f90. These optimizing and parallelizing compilers can generate 64-bit and 32-bit code. Compiling a FORTRAN program on the SGI Altix 3700 is similar to compiling a program on a typical UNIX system.

ifort -o prog prog.f

This command creates an executable called `prog`. The program is run by typing the program name at the system prompt.

./prog

Further optimization is available through the use of compiler flags and compiler directives. Please check the ***ifort*** and ***g77*** man pages for more details.

3.4 C/C++ Programming

The Intel C and C++ compilers are available on the SGI Altix 3700. These compilers are capable of optimizing and parallelizing code. Compiling a C program on the SGI Altix 3700 is similar to compiling a C program on a typical UNIX system.

icc -o prog prog.c

Compiling a C++ program is also just as similar.

icpc -o prog prog.cpp

These commands will create an executable program in a file called ***prog***. The program is executed by entering

./prog

Further optimization is available through the use of compiler flags and compiler directives. Please consult the icc or icpc man pages for more details.

3.5 Libraries

3.5.1 Math and Science Libraries

The SGI Altix 3700 has SCSL, a collection of mathematical and scientific libraries including BLAS levels 1, 2, and 3; LAPACK, a collection of solvers for dense linear algebra problems, including linear equations, linear least squares problems, eigenvalue problems, and singular value decomposition problems; Fast Fourier Transforms (FFTs); and convolutions. Both single-threaded and multi-threaded routines are available and select routines have been highly optimized to greatly improve performance. Users should use these library routines whenever possible.

This library is not automatically included in the link path. The user must specify the library when linking as in the following examples.

ifort -o prog prog.f -lscs
icc -o prog prog.c -lscs
icpc -o prog prog.c -lscs

Please consult the ***scsl*** man page for more details.

4. Running Jobs

4.1 Interactive Use

Interactive use is allowed, particularly for program development, including debugging and performance improvement, job preparation, job submission, and the preprocessing and postprocessing of data. However, only one node on the system is available for interactive use and interactive jobs are limited to 4 CPUs with 15 minutes of CPU time per process. Jobs with larger resource requirements must be submitted to the batch queues.

4.2 Batch Use

Load Sharing Facility (LSF), a networked subsystem for submitting, monitoring, and controlling a work load of batch jobs on one or more systems, is the batch system for the system. It provides services to monitor queue activity and to delete queued or running jobs. In the event of an orderly system shutdown, LSF jobs will be rerun from the beginning of the job (unless they are specifically marked not to be rerun). More information about LSF is available at

```
http://www.platform.com/products/HPC/
```

To allow users to run longer in the queues, a 2 week (336 hour) queue has been implemented, however, to keep the queue structure fair to all users, several restrictions have been put into place:

- The primary resource to schedule jobs by is the cpu hour (CPH). This quantity is equal to the wall time requested multiplied by the number of cpus and cannot exceed a value of 50,000 per user.
  (ncpus * walltime) = CPH

Table 1: Example CPH Values

# of Jobs	# of CPUs	Walltime	Total CPH
1	1000	48	48,000
2	256	48	24,576
25	8	96	19,200

- A user can use, at most, 1000 CPUs for MPI jobs and 500 CPUs for SMP and MPI/OpenMP hybrid jobs.
- The maximum wall time of any queue shall not exceed 336 hours.
- A background job cannot start if there is a foreground job in any queue.

The list of queues and the upper limits of job resources for these queues are available on the web at

```
http://www.asc.hpc.mil/overall/policy_procedure/policies/
use_policy.php
```

These limits are subject to change based on periodic review of system utilization and system configuration.

TIP: Because of CPH, it is not recommended that users accept the queue default walltime. If your job requires less time to run than the queue default, requesting the smaller of the two will result in a lower CPH and will allow you to **run more jobs** and **reduce your queue wait time**.

Example: User Joe submits a 4 CPU job using the queue default walltime of 336 hours. This results in a CPH of 1344. User Bob submits the same 4 CPU job, but only requests the 120 hours he needs to finish the job. His CPH would be 480. This allows user Bob to submit twice (2.8 to be exact) as many jobs as Joe in the same amount of CPH. If user Bob can alter his walltime and fit his jobs within 110 hours, he can submit 3 times the number of jobs as Joe. This also has a domino effect on the queue structure, resulting in faster throughput throughout the entire system.

4.2.1 Queueing Structure

The ASC MSRC SGI Altix 3700 has three queues: debug, regular, and background. These queues are available 24 hours a day, 7 days a week.

The debug queue accepts jobs that require up to 32 CPUs, 1 hour of CPU time, and 28 GB of memory. This queue is intended for short runs.

The regular queue is available for production work. Jobs that are submitted without any queue specified will go to the regular queue. The regular queue is divided into subqueues, but users do not submit jobs directly to these subqueues. Rather, the user specifies the number of CPUs, the CPU time, and memory requirements using the `bsub` options below. The job is then routed to the appropriate subqueue.

The background queue is also available for production work. Jobs run in the background queue are not charged against a user's allocation. However, jobs in the background queue are only started when utilization of the machine is low and never when foreground jobs are waiting.

The list of queues and the upper limits of job resources for these queues are available on the web at

```
http://www.asc.hpc.mil/overall/policy_procedure/policies/
batchqueue.php
```

These limits are subject to change based on periodic review of system utilization and system configuration.

4.2.2 Preparing Jobs

Before a user submits a job, they should prepare a job script. A job script is a UNIX shell script that contains all the commands the user will execute during the job. LSF will place the error and output files in the directory the job was submitted from, so scripts must be written with this in mind. Here is a sample job script.

```
#
#Change to WORK_DIR directory and copy input file.
#
cd $WORK_DIR
archive get -C {directory in $ARC} {filename}
#
#Run the analysis.
#
{My Program}
#
#Archive output and remove $WORK_DIR
#
tar cvf ../{output filename}.tar .
archive put -C {directory in $ARC} ../{output filename}.tar
rm -rf $WORK_DIR
#
#Exit the script.
#
exit
```

This script copies an input file and a program to the user's $WORK_DIR directory. The $WORK_DIR directory is a directory created by LSF for users to run their batch jobs. This directory has certain protections from the workspace scrubber, as long as the job is running, plus five days after it finishes, this directory will not be removed. Then the script changes to the $WORK_DIR directory, runs the program, copies two output files to permanent storage (one to the $HOME directory, one to the archival storage system $ARC), and then deletes the remaining files, the $WORK_DIR directory and exits.

NOTE: $WORK_DIR only exists in LSF, you will not be able to change to that directory using the variable $WORK_DIR.

4.2.3 Submitting Jobs

Once a job script is prepared, the `bsub` command is used to submit the script to LSF. The command has the following syntax:

bsub* < *script

Some important LSF options used are (type `man bsub` for a complete list of options available):

-q *queue*	Specifies the name of the queue to which the job will be submitted. For a list of allowable queues, please see: http://www.asc.hpc.mil/overall/policy_procedure/policies/use_policy.php
-n *n*	Specifies the number of CPUs the job will use.
-W hh:mm	Specifies the time limit for the job in walltime. The time should be specified in the hh:mm format (e.g., 15:00). *This is a required field, there is no default.

-o *outfilename*	Standard output (stdout) for the job is written to *outfilename*. *If you do not specify an output filename, LSF emails the output to your ASC email account.
-e *errfilename*	Standard error (stderr) for the job is written to *errfilename*. The default name is *jobname*.e*nnn* where *nnn* is the LSF identifier. *If you do not specify an error filename, LSF emails the error information to your ASC email account.
-J jobname	Specifies the name of the job.
-P account	Specifies the account number to charge to.
-a [SMP|MPI|MIX]	Specifies how jobs should be spread across nodes, if at all. SMP and MIX forces all CPUs to be allocated on a single node while MPI allows CPUs to be spread across multiple nodes. This is a required field.

When a job is submitted to LSF, a unique identifier is assigned to the job by the batch system similar to below:

```
2079.eagle-0.asc.hpc.mil
```

This identifier is needed when deleting a job.

Options of `bsub` commands are specified within the script file itself. The options are specified using syntax similar to PBS, but each line that contains an option must begin with the `#BSUB` string. Options that are specified within the script file must precede the first executable shell command of the file as in the following example.

```
#!/bin/csh
#BSUB -q regular
#BSUB -n 1
#BSUB -W 168:00
#BSUB -J test
#BSUB -o test.out
#BSUB -e test.out
#BSUB -a MPI
#BSUB -P WP+WPASC00000000**
```

**This is an example number. To find your account number, check your $ACCOUNT variable using

echo $ACCOUNT

More sample batch scripts can be found at the following URL:

```
http://www.asc.hpc.mil/customer/userdocs/samples/
samplebatch.php
```

4.2.4 Monitoring Jobs

The ***bjobs*** command is used to report the status of the batch jobs that are currently queued or running. Type `man bjobs` for information about `bjobs` and the options that are available.

The ***bjobs*** command lists all jobs that are running and queued.

bjobs -u all

```
JOBID   USER    STAT    QUEUE     FROM_HOST   EXEC_HOST   JOBNAME   SUBMIT_TIME
3373    user1   PEND    regular   eagle-0                 test      Feb 5 15:22
3971    user2   RUN     regular   eagle-0     eagle-3     test      Feb 5 15:22
```

Here is an explanation of the fields in the ***bjobs*** output.

Table 2: Fields from *bjobs*

Item	Meaning
JOBID	A unique identifier that consists of the original request number and the machine from which the request was submitted. Format is *nnn*, where *nnn* is an integer.
USER	Username of person submitting the job.
STAT	Job status. “RUN” indicates the job is running; “PEND” indicates the job is queued.
QUEUE	Name of the queue where the job is waiting or executing.
FROM_HOST	Cluster domain from which the job was submitted.
EXEC_HOST	Cluster domain where the job is running.
JOBNAME	Name of the job. This is either the name of the script file submitted to LSF or the name chosen with the -J flag.
SUBMIT_TIME	The date and time the job was submitted.

4.2.5 Deleting Jobs

In LSF, queued or running jobs are removed using the `bkill` command. The syntax is

bkill request-id

where *request-id* is the LSF identifier number.

Example:

bjobs

```
JOBID   USER   STAT    QUEUE     FROM_HOST   EXEC_HOST   JOBNAME   SUBMIT_TIME
3373    user   PEND    regular   eagle-0     eagle-3     test      Feb 5 15:22
```

bkill 3373

5. Customer Service

5.1 Customer Service Center

For customer assistance, call the ASC MSRC Service Center at 1-888-MSRC-ASC (1-888-677-2272), (937) 255-0194, or DSN 785-0194, or send e-mail with a description of the problem to msrchelp@asc.hpc.mil. The support analysts will help with anything related to ASC MSRC: third party software, UNIX, the different ASC MSRC computers, etc. If you have any questions about the ASC MSRC, contact the Service Center first. If your problem or question is beyond the scope of their expertise, they will refer you to the appropriate resource.

5.2 ASC MSRC Support

In-depth technical inquiries and problems are forwarded to the ASC MSRC Customer Assistance and Technology Center (CATC), which pursues such inquiries and problems through resolution as rapidly as possible. The ASC MSRC CATC will attempt to determine the nature of the problem, then identify and coordinate whatever resources are needed to resolve the problem.

The ASC MSRC also offers training classes, which provide an introduction to UNIX and the ASC MSRC. Intermediate and advanced classes on selected topics are also periodically announced on the Programming Environment and Training (PET) section of the ASC MSRC homepage. Topics for such classes may be requested through the Customer Service Center.

The ASC MSRC CATC is ready to support in an advisory capacity any engineer or scientist who is (or potentially is) an ASC MSRC user.

5.3 ASC MSRC Website

The ASC MSRC website is the best source for current ASC MSRC information. To access the ASC MSRC website simply access this URL: http://www.asc.hpc.mil.

Some of the topics found on the website include:

APPLICATIONS
Short and long descriptions of current ASC MSRC applications

`http://www.asc.hpc.mil/software/`

SYSTEMS
Information on ASC MSRC servers and Archival Storage

`http://www.asc.hpc.mil/hardware/`

CUSTOMER SERVICE
Available Customer Services

`http://www.asc.hpc.mil/customer/`

ONLINE DOCUMENTATION
Listings of the ASC MSRC User Guides are available for viewing. Instructions are given on obtaining postscript versions.

`http://www.asc.hpc.mil/customer/userdocs/`

VISUALIZATION LAB INFORMATION

Current status and other information about the Visualization Lab.

`http://www.asc.hpc.mil/sciviz/`

TRAINING

Current course offerings and schedule

`https://okc.erdc.hpc.mil/index.jsp`

FREQUENTLY ASKED QUESTIONS

Submit questions and read about various topics (such as “Customizing Your Environment”)

`http://www.asc.hpc.mil`

POLICIES AND PROCEDURES

The latest policies regarding usage of the ASC MSRC resources.

`http://www.asc.hpc.mil/overall/policy_procedure/`

Appendix A. Usage Hints

The following are tips and hints for the effective use of the SGI Altix 3700.

A.1 Runtime Considerations

A.1.1 Batch use is recommended

The SGI Altix 3700 system allocates more resources to batch jobs than for interactive use. Users will obtain the best throughput for long running or large memory jobs by submitting jobs to the batch queues.

A.1.2 Request only the time and memory needed

When submitting a job, choose the smallest queue that accommodates the job's time and memory requirements. Jobs that request significantly more resources than are actually needed can result in longer wait times and inefficient use of the machine.

A.2 Files and Filespace

A.2.1 File Management in workspace

A file scrubber is used to automatically remove old files from workspace to prevent them from becoming filled. The policy for removing files from these filesystems is available on our website. However, you are encouraged to remove files from workspace when they are no longer needed. This will minimize the overhead needed to enforce this policy.

The workspace filesystem is not backed up. It is your responsibility to transfer files that need to be saved to a location that allows permanent storage. Two possibilities are your $HOME directory or $ARC. Due to space restrictions on your $HOME directory, it is highly recommended that you use $ARC for long-term file storage and backups.

A.2.2 Archival Storage

To provide long-term storage and archiving, the ASC MSRC provides an archival storage system that combines a large-capacity Serial ATA drive cache, local tape backup/storage and a remote disaster recovery site.

This area is referenced using the environment variable $ARC, is NFS mounted to the interactive node of eagle and can be accessed much like any other directory in a filesystem. However, to improve system performance and increase transfer speeds, $ARC is not NFS mounted to the batch nodes of eagle. To transfer files in the batch environment, you will need to use unkerberized `rcp` or the `archive` command.

/usr/bin/rcp ${msas}:/msas*/foo/bar $WRK/username

archive get archive_filename local_filename

archive put local_filename

For more details about the Archival Storage system, see the *Archival Storage User's Guide,* located at:

```
http://www.asc.hpc.mil/customer/userdocs/
```

A.2.3 Keep I/O local to the system

The workspace filesystems are local to the SGI Altix 3700 via cxfs. Although `$HOME` is NFS-mounted internally to the compute nodes, I/O access from workspace will be faster than from the HAFS.

Here is a sample script that copies two input files (one from the $HOME directory, one from the archival storage system) and a program to the user's workspace area, changes to workspace, runs the program, copies two output files (one to the $HOME directory, one to the archival storage system) and then deletes the remaining files.

```
cd $WRK
cp $HOME/small.input $WRK
archive get big.input $WRK
archive get prog $WRK
prog
archive put big.output
mv small.output $HOME
rm small.input big.input prog big.output
```

A.3 Helpful SGI Altix 3700-related Websites

A.3.1 SGI

```
http://www.sgi.com/
```